Andrew Brinker

17 November 2013

Developing an Efficient Algorithm for Nimber Calculation in Dots and Boxes

Abstract

The impartial game of Dots and Boxes is an important one within the field of Combinatorial Game Theory. It is a relatively simple children's game, but the complexity of the play space and wide variety of emergent structures makes it ripe for combinatorial analysis. It also lacks a proven winning strategy, and includes game states with indeterminate optimal solutions.

As an impartial game, Dots and Boxes may be analyzed using the Sprague–Grundy theorem, allowing the application of nimber to positions within the game. In the context of Dots and Boxes these values allow for increased control of chains (a major emergent structural component of the play space). However, a naïve algorithm for nimber calculation runs in O(n!) time, making their calculation impractical until late game, when fewer potential moves are available and the usefulness of the counting itself is diminished.

In this paper I work to develop a more efficient algorithm, defined as one with a upper bound of $O(n^3)$ or better. I address certain potential methods for improving the efficiency of nimber calculations, and the proper data structures for both containing board states and operating on those states to find nimber. I will also discuss the application of a nimber calculation algorithm in the context of a general Dots and Boxes player AI.

Introduction

Dots and Boxes is a children's game with a simple set of rules. Two players play on an n × m—usually 5 × 5—grid of dots, taking turns drawing lines between the dots. When a player completes a unit square—a one-by-one square enclosure—by drawing the fourth line they receive a point and must draw another line. The game ends when all lines have been drawn and all boxes have been claimed, and the winner is the player who claimed the most boxes.

Dots and Boxes is an impartial game because at any given game state the moves available are independent of who is moving. It is popular with children because of the simplicity of its rules, but as discussed in the book Winning Ways for your Mathematical Plays, it is also NP-hard, and a particularly interesting game for analysis using combinatorial game theory (Berlekamp, Conway, and Guy).

The combination of the game's simple rules and open-ended play space results in a wide array of possible emergent structures as gameplay progresses. The game begins as an unconnected grid of dots, and it is the players themselves who, in drawing lines, provide structure to the game. The game can therefore be seen as a balancing act between the desires of the two players, who at any given point are both attempting to influence the composition of the game board in their favor.

There are four distinct levels of play in Dots and Boxes: random, greedy, double-dealing, and nim-counting. A player playing at one level is consistently able to defeat players playing at the lower levels, and transition between levels is not based on an increase in practice or skill, but on an improved conceptualization of the game's mechanics and underlying mathematics.

The first level of play, random, is the one most usually employed by small children, and is entirely devoid of strategy. Lines are drawn without reason, and boxes are claimed when available, but never explicitly sought out. The second level of play, greedy, is slightly better. Greedy players will avoid giving away lines to their opponents, and will take lines whenever they are available. This is the strategy most often used by adults playing the game, and is also the most obvious strategy available. It relies upon the assumption that the locally optimal choice—taking a box—is also the globally optimal one.

The third level of play, double-dealing, relies on an understanding of the necessity of sacrifice. As the game progresses the board often coalesces into chains of boxes arranged such that claiming one allows the player to immediately claim the others. These chains are central to the game's strategy, as they allow for a special type of move that may be exploited to maintain control of the board. The "double-deal" is a special move where the player sacrifices two boxes for the promise of more.

The idea works like this: imagine a board late in a game of Dots and Boxes. Most of the lines have been drawn, and all that's left are two long chains—unfilled boxes arranged so that claiming one allows the player to immediately claim the others—one of length 12, the other of length 13. Player B has just moved and setup player A to claim the shorter chain by drawing the third line on one of the chain's boxes. Player A has a choice: if they claim the whole chain they will be forced to setup B to claim the other chain, thereby losing; if they claim all but the last two boxes of the chain, player B gets two boxes, but gives Player A the second chain. Clearly, it is in Player A's best interest to claim 10 out of the 12 boxes in the first chain, leaving the final two

positioned so that they both have three lines drawn and the fourth line needed is shared between them—the eponymous "double deal”.

The most interesting strategic strength of the double deal strategy is that the player being double-dealt has no incentive not to take the double deal. In the example given above, if Player B chooses not to take the two boxes they've been offered, their only other choice is to setup the second chain, thereby giving all both the two boxes and the chain to Player A.

The difficulty of the double-deal strategy is that it does not prescribe a methodology for ending up in a position where you can offer a double-deal, it merely prescribes what to do once you are in such a position. This leaves room for the fourth strategy, nim-counting, which is solely concerned with chain control. Before this strategy can be discussed though, it is important to discuss the game Nim.

Nim is a classic impartial game where two players are presented with a collection of heaps of objects (Ferguson). There can be any number of heaps, not necessarily of the same size and each with any number of objects in them. The players take turns removing items from the heaps, and the player to remove the last item from the last heap wins—called the "normal play condition." The difficulty of Nim comes from the condition on moves: in any given turn objects may not be removed from more than one heap, although any number of objects may be removed from any one heap.

Nim has a known optimal strategy, based on a type of number called "nim-values" or "nimbers". Nimbers are the values of nim-heaps, and are added using nimber addiction, which can be simply defined as the exclusive-or of two nimbers. In Nim, if the nim-sum of the nimber of each heap is zero, the first player to move will win. If the nim-sum is not zero, the second

player to move will win. For whomever will win based on the above rule, the winning strategy for the given player is to remove objects from heaps so that at the end their own turn the nim-sum of all the heaps is zero.

In the 1930's, R.P. Sprague and P.M. Grundy independently proved that any independent game under the normal play condition could be generalized to a game of Nim (Milvang-Jensen). This means that the optimal strategy for those games is then an analogue of the optimal strategy for nim: making sure the nim-sum of the game's heaps is zero at the end of your own turn.

You may be wondering how this is relevant to Dots and Boxes, considering that Dots and Boxes does not follow the normal play condition. Well, in the book The Dots and Boxes Game: Sophisticated Child's Play Elwyn Berlekamp proved that the Sprague-Grundy Theorem may be applied to Dots and Boxes, with the nimber of a position being the minimum excluded value—defined as the smallest integer not present in the set—of all following positions, and the heaps being regions that develop as the game progresses.

To see how the theorem may be applied to Dots and Boxes, consider that under the double-dealing strategy, if we assume optimal play, the outcome of the game is decided once the board has been reduced to nothing but chains. This means that until that point the result of the game is not decided, and that the goal is then to be at the proper movement position (either currently moving or being next to move) once the board is reduced to chains. Clearly, this follows the normal play condition, and so the Sprague-Grundy Theorem may be applied to Dots and Boxes.

To see how useful this application is, consider a 5×5 board with a single chain and two distinct regions—distinct meaning that the two regions are separates by a complete wall of lines.

Each of the two regions is largely undecided, with only one or two lines drawn in them. You want to guarantee that you can claim the chain later in the game. Without the Sprague-Grundy Theorem you would be forced to leave that control largely to chance, but with the theorem you can calculate the nimber of each of the two regions, and then make sure that the nim-sum of the two regions is 0 at the end of your turns, same as in Nim. By following this rule, you can guarantee control of the chain.

Discussion

Given this, it becomes important that players be able to quickly determine what the nimber of a position is. In Nim the nimber of a position was simply the size of the heap, but in Dots and Boxes calculating the value is not so simple. The nimber of a position in Dots and Boxes is defined as the mex of the nimber of all following positions, with positions where all follower moves are loony—meaning they result in either player gaining a point—having a nimber of 0 (Canard). Here is that algorithm given in pseudocode:

```
def mex(values):
      counter = 0
      for value in values:
            if counter != value:
                  return counter
            counter += 1
      return values[len(values) - 1] + 1
def naive_nim(position):
      if all following moves for position are loony:
            return 0
      return mex(all following positions to position)
```

The algorithmic complexity of this algorithm is O(n!). To see this, consider that the input size to the algorithm, n, is the number of lines not yet filled in the current region. This means that

there are n following positions to the current one. For each of those positions there are then n - 1 lines not yet drawn, meaning that each position has n - 1 follower positions. This logic can be applied recursively, leading to the O(n!) complexity.

The inefficiency of this algorithm severely hinders its usefulness. It also requires recalculation of nimbers after each move, because the nim value of each following position is not saved. This means that you'll be running the entire algorithm at most n more times, making the runtime even worse than O(n!) at O(n • n!).

Developing a more efficient algorithm is clearly important for the development of a player AI that can play at the nim-counting level. The first step to improving efficiency is memoization—the saving of calculated values to avoid recalculation.

To see the usefulness of memoization, consider the following: there is a position with two lines missing, line A and line B. The two potential following positions are therefore the position where line A is chosen, and the position where line B is chosen. However, the only follower position for each of those positions is where the other line is chosen. Without memoization that position would have its nimber calculated twice. With memoization it will only be calculated once.

However, this approach still seems backwards, and can be better conceptualized working in reverse. For a region R, create a filled board B that is a container of minimum size for R—to accommodate irregular regions. Calculate the nimber of all positions directly preceding the current state of B—in other words, all positions with 1 line removed, or all followers of R with n - 1 lines added. Save the values as they are calculated. Then calculate the nimber of each position with 2 lines removed. Repeat until the number of lines removed is equal to n. Make sure to only

remove lines not in R. This conceptualization makes the fact that states will not be recalculated clear and easier to reason about. Here is some very rough pseudocode explaining this procedure:

```
function fast_nim(region):
      board = board with the same dimensions as region
      l = number of missing lines in region
      n = 1
      saved = set of positions w/ 1 line removed, each with value 0
      while (n != l):
            G = set of antecedent positions with n lines removed
            for each position in G:
                  F = set of follower positions to position
                        nim_values = []
                        for each follower in F:
                              {follower position is always in saved}
                              nim_values = saved[follower]
                        saved[position] = mex(nim_values)
      return saved
```

Unlike the naive algorithm, the above algorithm runs in $O(n^3)$ time. The first loop removes successively large numbers of lines. The second iterates through each of the resulting positions. The third calculates the nimber of each position using the previously saved values. At the end, return the full dictionary of positions and nimbers, which can then be searched following each move to avoid recalculation.

As you can see, this algorithm is significantly more efficient than the naïve one. This means that a computer player should be reasonably able to use this algorithm to calculate nimber, and then use those values to adjust strategy according to the previously discussed methods.

Conclusion

This algorithm does not prescribe when to begin performing nimber calculations, nor does it prescribe how those calculations should be used once they are performed, but it does provide a more efficient method for calculating the nimber of a region, and should greatly simplify the development of an advanced Dots and Boxes player AI.

There are still potential avenues for improvement in the algorithm. The nimber of a position is equal to the nimber of any rotation, and so once a position's nimber has been calculated one should be able to avoid calculating the nimber of any rotation. The difficulty in such an advancement is in identifying whether two given positions are rotations, which is a specialized version of determining whether the two graphs—the Dots and Boxes board is a graph, after all—are isomorphic.

It is entirely possible that $O(n^3)$ is not the fastest a nimber calculation algorithm can run. However, such a running time is significantly better than the naïve algorithm's O(n!) (or O(n • n!) with repeated uses). This makes the calculation of nimbers by a Dots and Boxes AI player more reasonable, and should greatly assist in developing an AI player that can consistently play at the nim-counting level.

Works Cited

Berlekamp, Elywn. John Conway. Richard Guy. *Winning Ways for Your Mathematical Plays*. New York: New York Academic Press. 1982. Print.

Berlekamp, Elwyn. *The Dots and Boxes Game: Sophisticated Child's Play*. Natick, Massachusetts: A K Peters, Ltd. 2000. Print.

Canard, W. C. "Nim-Theory in the Game of Dots and Boxes." Paintserv.us. Paintserv. September 27, 2007. Web. October 15, 2013.

Ferguson, Thomas. "Game Theory." Los Angeles: UCLA Mathematics Department, n.d. Print.

Milvang-Jensen, Brit C. A. "Combinatorial Games, Theory and Applications." Copenhagen: University of Copenhagen, 2000. Print.